P2 calculate mean median and mode of a list of a number implement basic stastical calcluation usi scala collection

Built.sbt

```
val scala3Version = "3.7.3"


lazy val root = project
  .in(file("."))
  .settings(
    name := "p2",
    version := "0.1.0-SNAPSHOT",


    scalaVersion := scala3Version,


    libraryDependencies += "org.scalameta" %% "munit" % "1.0.0" % Test
  )
```

Main.scala

```
object StatisticsCalculator {
  def main(args: Array[String]): Unit = {
    val nums = List(4, 5, 6, 2, 4, 4, 7, 5, 2)


    def mean: Double = nums.sum.toDouble / nums.size


    def median: Double = {
      val s = nums.sorted
      val n = s.size
      if (n % 2 == 1) s(n / 2)
      else (s(n / 2 - 1) + s(n / 2)) / 2.0
```

```
    }

    def mode: List[Int] = {
      val freq = nums.groupBy(identity).view.mapValues(_.size).toMap
      val maxFreq = freq.values.max
      freq.filter(_._2 == maxFreq).keys.toList
    }

    println(s"Numbers: $nums")
    println(f"Mean: $mean%.2f, Median: $median%.2f, Mode:
${mode.mkString(", ")}")
  }
}
```

P3 genarte a random data set of a 10 numbers and calcluate its variance and std deviation

Built.sbt

```
val scala3Version = "3.7.3"

lazy val root = project
  .in(file("."))
  .settings(
    name := "p3",
    version := "0.1.0-SNAPSHOT",

    scalaVersion := scala3Version,

    libraryDependencies += "org.scalameta" %% "munit" % "1.0.0" % Test
```

```
  )
```

Main.scala

```
import scala.util.Random
import scala.math.sqrt

object VarianceStdDevCalculator {
  def main(args: Array[String]): Unit = {
    val data = List.fill(10)(Random.nextInt(100) + 1)

    val mean = data.sum.toDouble / data.size
    val variance = data.map(x => math.pow(x - mean, 2)).sum / data.size
    val stdDev = sqrt(variance)

    println(s"Data: $data")
    println(f"Mean: $mean%.2f, Variance: $variance%.2f, Std Dev:
$stdDev%.2f")
  }
}
```

P4 create a dense vector using breeze and calcluate its sum mean and dot product with a vector

Built.sbt

```
val scala3Version = "3.7.3"
libraryDependencies += "org.scalanlp" %% "breeze" % "2.1.0"
lazy val root = project
  .in(file("."))
  .settings(
    name := "p4",
```

```
    version := "0.1.0-SNAPSHOT",

    scalaVersion := scala3Version,

    libraryDependencies += "org.scalameta" %% "munit" % "1.0.0" % Test
  )
```

Main.scala

```
import breeze.linalg._
import breeze.stats._

object MovingAverage {
  def main(args: Array[String]): Unit = {
    val v1 = DenseVector(1.0, 2.0, 3.0, 4.0, 5.0)
    val v2 = DenseVector(5.0, 4.0, 3.0, 2.0, 1.0)

    println(s"v1: $v1\nv2: $v2")
    println(f"Sum: ${sum(v1)}%.2f, Mean: ${mean(v1)}%.2f, Dot Product:
${v1 dot v2}%.2f")
  }
}
```

P5 generate a random matrix using breeze and compute its tranpose and determinat

Built.sbt

```
val scala3Version = "3.7.3"
libraryDependencies += "org.scalanlp" %% "breeze" % "2.1.0"
```

```
lazy val root = project
  .in(file("."))
  .settings(
    name := "p4",
    version := "0.1.0-SNAPSHOT",

    scalaVersion := scala3Version,

    libraryDependencies += "org.scalameta" %% "munit" % "1.0.0" % Test
  )
```

Main.scala

```
import breeze.linalg._
import breeze.stats.distributions._
import breeze.stats.distributions.Rand.FixedSeed.randBasis

object BreezeMat {
  def main(args: Array[String]): Unit = {
    // Provide the implicit RandBasis
    implicit val basis: RandBasis = randBasis

    // Generate a 3x3 matrix with Uniform(0,10) values
    val mat = DenseMatrix.rand(3, 3, Uniform(0, 10))
    println("Matrix:\n" + mat)
    println("\nTranspose:\n" + mat.t)
```

```
    // Calculate determinant
    val determinant = det(mat.map(_.toDouble))
    println(f"\nDeterminant: $determinant%.4f")
  }
}
```

P6 slice a breeze matrix to extract sum matrix and calcluate its row and column sums

Built.sbt

```
val scala3Version = "3.7.1"

lazy val root = (project in file("."))
  .settings(
    name := "breezeslice",
    version := "0.1.0-SNAPSHOT",
    scalaVersion := scala3Version,

    libraryDependencies ++= Seq(
      "org.scalanlp" %% "breeze" % "2.1.0",
      "org.scalameta" %% "munit" % "1.0.0" % Test
    )
  )
```

Main.scala

```
import breeze.linalg._
```

```
object BreezeSlice {
  def main(args: Array[String]): Unit = {
    val matrix = DenseMatrix(
      (1.0, 2.0, 3.0, 4.0),
      (5.0, 6.0, 7.0, 8.0),
      (9.0, 10.0, 11.0, 12.0)
    )

    val subMatrix = matrix(1 to 2, 1 to 2)
    println(s"Sub-matrix:\n$subMatrix")

    val rowSums = sum(subMatrix(*, ::))
    val colSums = sum(subMatrix(::, *))

    println(s"Row sums: $rowSums")
    println(s"Column sums: $colSums")
  }
}
```

P7 write a program to perform element wise addition , subtraction, multiplication , division of two breeze matrix

Built.sbt

```
val scala3Version = "3.7.1"

lazy val root = (project in file("."))
```

```
  .settings(
    name := "breezeslice",
    version := "0.1.0-SNAPSHOT",
    scalaVersion := scala3Version,
    libraryDependencies ++= Seq(
      "org.scalanlp" %% "breeze" % "2.1.0",
      "org.scalameta" %% "munit" % "1.0.0" % Test
    )
  )
```

Main.scala

```
import breeze.linalg._

object BreezeSlice {
  def main(args: Array[String]): Unit = {
    val matrixA = DenseMatrix(
      (1.0, 2.0, 3.0),
      (4.0, 5.0, 6.0),
      (7.0, 8.0, 9.0)
    )

    val matrixB = DenseMatrix(
      (9.0, 8.0, 7.0),
      (6.0, 5.0, 4.0),
      (3.0, 2.0, 1.0)
```

```
    )

    val addition = matrixA + matrixB
    val subtraction = matrixA - matrixB
    val multiplication = matrixA *:* matrixB   // Element-wise multiplication
    val division = matrixA /:/ matrixB         // Element-wise division

    println("Addition:\n" + addition)
    println("Subtraction:\n" + subtraction)
    println("Multiplication (element-wise):\n" + multiplication)
    println("Division (element-wise):\n" + division)
  }
}
```

P11 write a program to tokenzie and count the frequency of word in the text file

Built.sbt

```
val scala3Version = "3.7.3"

lazy val root = project
  .in(file("."))
  .settings(
    name := "p11",
    version := "0.1.0-SNAPSHOT",
```

```
    scalaVersion := scala3Version,

    libraryDependencies += "org.scalameta" %% "munit" %
"1.0.0" % Test
  )
```

Sample.txt

```
Scala is fun.
Scala is powerful.
Learn Scala programming.
```

Main.scala

```
import scala.io.Source

object WordFrequencyCounter {
  def main(args: Array[String]): Unit = {
    val words = Source.fromFile("sample.txt")
      .getLines()
      .mkString(" ")
      .toLowerCase
      .split("\\W+")
      .filter(_.nonEmpty)

    val freq = words
      .groupBy(identity)
```

```
      .view
      .mapValues(_.size)
      .toSeq
      .sortBy(-_._2)

    freq.foreach { case (word, count) =>
      println(s"$word\t$count")
    }
  }
}
```